AWS S3 Buckets + Lambda + API Gateway + Confluence Integration

This project demonstrates how to create multiple S3 buckets across regions, use a Lambda function to fetch their names and tags, expose the Lambda via API Gateway, and display the results dynamically on a Confluence page using an HTML macro.

# Steps to Implement

## 1. Create S3 Buckets with Tags

1. Go to the AWS Management Console → S3 → Create bucket.  
2. Enter a unique bucket name (e.g., your-company-platforms-mum-001).  
3. Select a region (e.g., ap-south-1, eu-west-1, us-east-1).  
4. Under Tags, add: Key = Owner, Value = Platforms.  
5. Create the bucket and repeat for 3 different regions.

## 2. Create IAM Role for Lambda

1. Go to IAM → Roles → Create role.  
2. Select AWS Service → Lambda.  
3. Attach policies: AmazonS3ReadOnlyAccess, AWSLambdaBasicExecutionRole.  
4. Name it s3-list-buckets-role.  
5. Create the role.

## 3. Create Lambda Function

1. Go to Lambda → Create function.  
2. Runtime: Python 3.12, Architecture: arm64.  
3. Use existing role: s3-list-buckets-role.  
4. Paste the following code:

import json  
import boto3  
from concurrent.futures import ThreadPoolExecutor, as\_completed  
  
s3 = boto3.client("s3")  
  
def get\_owner(bucket):  
 try:  
 tags = s3.get\_bucket\_tagging(Bucket=bucket)  
 for t in tags.get("TagSet", []):  
 if t["Key"].lower() == "owner":  
 return t["Value"]  
 except:  
 return "—"  
 return "—"  
  
def lambda\_handler(event, context):  
 resp = s3.list\_buckets()  
 names = [b["Name"] for b in resp.get("Buckets", [])]  
   
 results = []  
 with ThreadPoolExecutor(max\_workers=10) as executor:  
 futures = {executor.submit(get\_owner, n): n for n in names}  
 for f in as\_completed(futures):  
 results.append({"name": futures[f], "owner": f.result()})  
   
 return {  
 "statusCode": 200,  
 "headers": {"Content-Type": "application/json", "Access-Control-Allow-Origin": "\*"},  
 "body": json.dumps({"buckets": results})  
 }

## 4. Create API Gateway

1. Go to API Gateway → Create API → HTTP API.  
2. Name: BucketsAPI.  
3. Add integration: Lambda function (list-buckets-fast).  
4. Create route: GET /buckets.  
5. Deploy with default stage.  
6. Copy the Invoke URL (e.g., https://abc123.execute-api.ap-south-1.amazonaws.com/buckets).

## 5. Confluence Integration

1. Enable or install an HTML macro (depending on Confluence version).  
2. Edit a Confluence page → Insert HTML Macro.  
3. Paste the following code:

<div id="s3-buckets-root">Loading buckets…</div>  
<script>  
(async () => {  
 const API = "https://abc123.execute-api.ap-south-1.amazonaws.com/buckets";   
 const root = document.getElementById("s3-buckets-root");  
 try {  
 const res = await fetch(API);  
 const data = await res.json();  
 let html = `<table border="1" cellpadding="6" style="border-collapse:collapse;">  
 <tr><th>Name of Bucket</th><th>Owner</th></tr>`;  
 data.buckets.forEach(b => {  
 html += `<tr><td>${b.name}</td><td>${b.owner}</td></tr>`;  
 });  
 html += `</table>`;  
 root.innerHTML = html;  
 } catch (e) {  
 root.innerHTML = "Error loading buckets.";  
 }  
})();  
</script>

# Benefits

- Centralized monitoring of S3 buckets.  
- Real-time display of bucket names and owners in Confluence.  
- Serverless, low-cost, and highly scalable.  
- Easy to extend to 7 or more buckets.